Methods in planning
A historic review

Batian Niebel

May 3, 2017

Introduction Interest in using intelligent algorithms for solving planning problems arose with the advent of computers. Linear programming, which can also be seen as planning under linear constraints, was used heavily during the Second World War [10]. It allowed developing optimal plans for resource allocation. But the shortcomings imposed by linear programming, only linear constraints and continuous variables, lead to the development of algorithms that overcame these burdens. These algorithms put planning to the next level. These algorithms allowed for logical constraints, discontinuous states spaces, infinite planning horizons and probabilistic effects. Here, the major result of planning research are briefly reviewed.

Direct state search First planning algorithms were state-space search algorithms. Breadth first search can be traced back to dynamic programming [1]. Uniformed cost search to the Dijkstra's shortest path algorithm [3]. These algorithms, being uninformed in nature, all suffered from the curse of dimensionality. Considering the computers of the 1960's this was a huge burden for using those algorithms at this time.

A* search [6], which used a heuristic function to approximate the distance to the goal, proved very successful in overcoming the limitations of uninformed search algorithms. But coming up with a good heuristics required human intelligence. Thus the next improvement of artificial intelligence was the development of algorithms that automatically device heuristics for the use in A* search and similar algorithms.

Planning systems In the 1970's, one of the first planning systems, Strips, introduced a formal domain independent language for the description of planning problems [5]. Strips was already remarkably close to PDDL [11], a planning problem description language from the 1990's. Theses planning languages provided the basis for algorithms that exploited the formal abstraction for the design of automatic planning heuristics.

Satplan transforms the planning problem into boolean satisfiability problems [8], which are also commonly solved in circuit design or automatic theorem proving. This satisfiability problem is then solved algorithms such as the Davis-Putnam-Logemann-Loveland algorithm [4]. Graphplan iteratively expands the planning problem into a directed graph, where the nodes are actions and atomic facts (states)[2]. By introducing constraints which exclude mutually exclusive nodes, Graphplan expands the state space into the direction of possible solutions, thus allowing for planning in large state spaces.

Other planners Another field of planning is the usage of boolean algebra. Here, also integer programming can be used, providing excellent algorithms for complex and large state spaces, such as IBM's ILog solvers [7]. Markov decision processes allow developing planning algorithms considering uncertainty [9].

Conclusion & outlook Although 60 years old, planning is a very active field in artificial intelligence. The description of complex problems in very defined high level languages, such as PDDL, allows to create very efficient algorithms to automatically find solutions in extremely large state spaces. It will be interesting to see, how planning will benefit, and also fuel methods such as deep neural nets and deep reinforcement learning in the near future.

References

[1] Richard Bellman. "Dynamic programming and Lagrange multipliers". In: *Proceedings of the National Academy of Sciences* 42.10 (1956), pp. 767–769. URL: http://www.pnas.org/content/42/10/767.short (visited on 05/03/2017).

[2] Avrim L. Blum and Merrick L. Furst. “Fast planning through planning graph analysis”. In: *Artificial Intelligence* 90.1 (Feb. 1, 1997), pp. 281–300. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(96)00047-1. URL: http://www.sciencedirect.com/science/article/pii/S0004370296000471 (visited on 05/01/2017).

[3] Thomas H. Cormen et al. *Introduction to algorithms second edition*. The MIT Press, 2001. URL: http://www.eecis.udel.edu/~lliao/cis320f05/lec11.pdf (visited on 05/03/2017).

[4] Martin Davis, George Logemann, and Donald Loveland. “A machine program for theorem-proving”. In: *Communications of the ACM* 5.7 (1962), pp. 394–397. URL: http://dl.acm.org/citation.cfm?id=368557 (visited on 05/03/2017).

[5] Richard E. Fikes and Nils J. Nilsson. “Strips: A new approach to the application of theorem proving to problem solving”. In: *Artificial Intelligence* 2.3 (Dec. 1, 1971), pp. 189–208. ISSN: 0004-3702. DOI: 10.1016/0004-3702(71)90010-5. URL: http://www.sciencedirect.com/science/article/pii/0004370271900105 (visited on 05/03/2017).

[6] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. “A formal basis for the heuristic determination of minimum cost paths”. In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. URL: http://ieeexplore.ieee.org/abstract/document/4082128/ (visited on 05/03/2017).

[7] *IBM CPLEX CP Optimizer*. URL: https://www-01.ibm.com/software/commerce/optimization/cplex-cp-optimizer/index.html (visited on 05/03/2017).

[8] Henry A. Kautz, Bart Selman, et al. “Planning as Satisfiability.” In: *ECAI*. Vol. 92. Citeseer, 1992, pp. 359–363. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.9443&rep=rep1&type=pdf (visited on 05/03/2017).

[9] Andrey Kolobov. “Planning with Markov decision processes: An AI perspective”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012), pp. 1–210. URL: http://www.morganclaypool.com/doi/abs/10.2200/S00426ED1V01Y201206AIM017 (visited on 05/03/2017).

[10] *linear programming — mathematics*. Encyclopedia Britannica. URL: https://www.britannica.com/topic/linear-programming-mathematics (visited on 05/01/2017).

[11] Drew McDermott et al. “PDDL-the planning domain definition language”. In: (1998). URL: http://www.citeulike.org/group/13785/article/4097279 (visited on 05/03/2017).